AI Bootcamp Lab Project – Fire Risk Prediction System

Welcome to your AI Bootcamp Lab! In this lab, you'll apply your skills in Linear Regression, Multiple Variable Regression (using Gradient Descent and scikit-learn), and Logistic Regression to a high-impact real-world problem: predicting fire risk in smart cities.

# Project Title: Early Fire Risk and Severity Prediction in Smart Cities

Your task is to build a predictive AI system that can:  
1. Estimate a continuous Fire Risk Score (0-100) for urban areas based on environmental and sensor data.  
2. Classify each location into risk levels: Low, Medium, or High.  
This dual-task setup lets you explore both regression and classification in a real-world context.

# Dataset Overview

You are provided with two datasets:  
- fire\_risk\_train.csv – includes features and labels for training.  
- fire\_risk\_test.csv – includes features only (for testing your model).  
- fire\_risk\_submission\_template.csv – format to prepare your predictions for evaluation.  
  
Features include:  
• Temperature (°C)  
• Humidity (%)  
• Wind Speed (km/h)  
• Air Quality Index (AQI)  
• Vegetation Index (0 to 1)  
• Emergency Response Time (minutes)  
• A noisy feature (random\_noise) – used for feature selection challenge.  
  
Target Columns (in training data):  
- fire\_risk\_score (Regression Target)  
- fire\_risk\_level (Classification Target: Low, Medium, High)

# Tasks

Your work is divided into the following stages:

* 1. Exploratory Data Analysis (EDA)
* 2. Implement Linear Regression to predict `fire\_risk\_score`
* 3. Build a Multiple Variable Regression model using scikit-learn
* 4. Implement custom Gradient Descent (optional bonus)
* 5. Convert score to labels (Low/Medium/High) using thresholds
* 6. Train Logistic Regression to classify `fire\_risk\_level`
* 7. Prepare your predictions for test data and save to CSV

# Evaluation

Evaluation will be based on the following:  
- RMSE for `fire\_risk\_score` predictions  
- Accuracy or F1 Score for `fire\_risk\_level` classification  
- Code readability and modularity  
- Proper feature scaling and feature selection  
- Bonus: Manual implementation of Gradient Descent

# Notes

- Don’t forget to investigate correlations and drop irrelevant/noisy features (like random\_noise).  
- Normalize/standardize your data as needed.  
- You may use matplotlib/seaborn for EDA visualizations.  
- Ensure your code is commented and neatly structured.

# Submission Instructions

Submit the following:  
1. Jupyter/Colab Notebook with code and outputs.  
2. Final CSV predictions (on test set).  
3. A short PDF report or markdown in notebook describing your models, findings, and results.

Happy Coding & Stay Safe from Fires!